# Brief Article

The Author

April 19, 2016

## 1 Mathematische Grundlagen

Zunächst einmal beschäftigen wir uns mit den mathematischen Grundlagen die für eine Betrachtung des Diskreten Logarithmus-Problems unabdingbar sind.

### 1.1 Gruppen

Es sei $(G, \Delta)$ eine Gruppe wie wir sie bereits aus anderen Vorlesungen kennen.
Die Ordnung der Gruppe entspricht der Anzahl der Elemente in G.
Die Ordnung eines Elements $\alpha \in G$ ist die Zahl $n$, für die gilt: $\alpha^n = e$, wobei $e$ das neutrale Element bzgl. $\cdot$ ist.
Es sei $n \in N$ und $\alpha^n = 1$ sowie $\alpha^{n/p} \neq 1$ für alle Primteiler $p$ von $n$. Dann ist $n$ die Ordnung von $\alpha$.
Die Gruppe heißt zyklisch, wenn ein $\alpha \in G$ existiert, so dass gilt:

$$\forall a \in G \; \exists i \in N : \alpha^i = \alpha$$

$\alpha$ heißt Generator der zyklischen Gruppe.
Sei $(G, \cdot\, p)$ eine Gruppe und $\alpha \in G$.
$\alpha$ habe eine endliche Ordnung. Man schreibt $\langle \alpha \rangle$ für die von $\alpha$ erzeugte Untergruppe.

### 1.2 Primwurzel

Es sei $(G, \cdot\, p)$ eine Gruppe, $\alpha \in G$ und $p = |G|$. $\alpha$ heißt Primitivwurzel, wenn $\alpha$ die Ordnung $p$ hat und $ggT(\alpha, p) = 1$.
Es sei $(Z_p, \cdot p)$ eine prime Restklassengruppe und $\alpha \in Z_p$.
Weiterhin seien $p_1, p_2, \cdots, p_n$ die Primfaktoren von $G$.
$\alpha$ ist eine Primitivwurzel, wenn für $1 \leq r \leq n$ gilt $\alpha^{pr} \neq 1$.

### 1.3 Beispiel

Als Beispiel dient die Primzahl p = 13 und die dazu gehörige prime Restklassengruppe $G = (Z/13Z)^{\times} = \{1, 2, \ldots, 12\}$. Zur Primitivwurzel $g = 2$ wird nun die Wertetabelle der diskreten Exponentiation bestimmt.

$$2^0 = 1 \equiv 1 \bmod 13$$
$$2^1 = 2 \equiv 2 \bmod 13$$
$$2^2 = 4 \equiv 4 \bmod 13$$
$$2^3 = 8 \equiv 8 \bmod 13$$
$$2^4 = 16 \equiv 3 \bmod 13$$
$$2^5 = 32 \equiv 6 \bmod 13$$
$$2^6 = 64 \equiv 12 \bmod 13$$
$$2^7 = 128 \equiv 11 \bmod 13$$
$$2^8 = 256 \equiv 9 \bmod 13$$
$$2^9 = 512 \equiv 5 \bmod 13$$
$$2^{10} = 1024 \equiv 10 \bmod 13$$
$$2^{11} = 2048 \equiv 7 \bmod 13$$

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^a \equiv mod 11$ | 1 | 2 | 4 | 8 | 3 | 6 | 12 | 11 | 9 | 5 | 10 | 7 |

Wie man sehen kann sind die Potenzen paarweise disjunkt und die gesamte prime Restklassengruppe kann mithilfe den Potenzen von g erstellt werden. Durch vertauschen und sortieren der Tabelle bekommt man die Wertetabelle für den diskreten Logarithmus.

| $x$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\log_2$ x | 1 | 2 | 5 | 3 | 10 | 6 | 12 | 4 | 9 | 11 | 8 | 7 |

### 1.4 Diskreter Logarithmus Problem

Voraussetzungen:

Sei $(G, \cdot\, p)$ eine multiplikative Gruppe, $\alpha \in G$ ein Element der Ordnung n und $\beta \in \langle\alpha\rangle$

**Problem:**
Man berechnet ein $a$ im Bereich $0 \leq a \leq n-1$, so dass $\alpha^a = \beta$ ist. a nennt man den diskreten Logarithmus von $\beta$ zur Basis $\alpha$.
Dieses einfach anzumutende Problem ist einer der großen Stützpfeiler, auf die sich unsere Kryptosysteme und Sicherheitssysteme stützen. Die Lösung dieses Problems bedarf im Vergleich zur Diskreten Exponentiation erheblich mehr Rechenaufwand. Die effiziente Lösung des diskreten Logarithmus ist eine Herausforderung der heutigen Mathematik und Kryptologie die uns vermutlich noch eine sehr lange Zeit beschäftigen wird.

## 1.5 Kleiner fermatscher Satz

Der kleine fermatsche Satz sagt aus, dass bei einer ganzen Zahl a und einer Primzahl p gilt:

$$a^p \equiv a \bmod p$$

Falls a kein Vielfaches von p ist, so kann man die Gleichung umformen:

$$a^{p-1} \equiv 1 \bmod p$$

### 1.5.1 Beweis des kleinen fermatschen Satzes

Wir wollen zeigen, dass für eine Primzahl p und eine beliebige ganze Zahl a gilt: $a^p \equiv a \bmod p$. Wenn man es umformuliert kann man hiermit auch sagen, dass $a^p - a$ durch p teilbar ist.

Ist a durch p teilbar, so gilt bereits $a \equiv 0 \equiv a^p \bmod p$.
Wir beweisen den kleinen fermatschen Satz durch Induktion.

*Induktionsanfang*: $0^p - 0 = 0$ und das ist durch p restlos teilbar.
*Induktionsschritt*: Die Behauptung sei wahr für ein bestimmtes a. Somit gilt für a+1:

$$(a+1)^p - (a+1) = a^p + \tbinom{p}{1} a^{p-1} + \cdots + \tbinom{p}{p-1} a + 1 - (a+1)$$

Wobei bei jedem Binominialkoeffizienten gilt, dass

$$\tbinom{p}{k} = \frac{p \cdot (p-1) \cdots (p-k+1)}{1 \cdot 2 \cdots k}$$

und damit auch dass p mit $1 \leq k \leq p-1$ nur im Zähler auftaucht. Da wir angenommen haben, dass p prim ist, tauchen auch sonst im Nenner keine weiteren Teiler auf. Die Binominialkoeffizienten sind demnach also alle durch p teilbar, da mindestens ein p im Zähler ist. Daraus folgt schließlich:

$$(a+1)^p - (a+1) \equiv a^p + 1 - (a+1) = a^p - a \mod p$$

und nach der Induktionsvoraussetzung ist dies durch p teilbar.

Zur Berechnung des Diskreten Logarithmus gibt es noch keine schnellen Algorithmen. Die diskrete Exponentialfunktion hingegen lässt sich in kürzester Zeit sehr einfach berechnen. Dies macht den diskreten Logarithmus zu einer perfekten Einwegfunktion die heute sehr weit in verschiedensten Kryptosystemen im Einsatz ist.
Beispiele hierfür sind der Diffie-Hellmann-Schlüsselaustausch, Elgamal-Schlüsselaustausch, welchen wir später noch kennene lernen und der Digitale Signatur Algorithmus. Ein weiterer Algorithmus ist das Massey-Omura-Schema, das wir nun ausführlicher behandeln.

# 2 Massey-Omura-Schema

Der Grundgedanke an die Anforderungen eines Sicheren Nachrichtensystems war, dass beide Parteien weder den privaten noch den öffentlichen Schlüssel der anderen Partei besitzen oder auf diesen Schlussfolgern können. Das Massey-Omura-Schema wurde hierzu im Jahre 1983 von den Kryptologen James Massey und Jim Omura entwickelt und basiert auf der Schwierigkeit, den diskreten Logarithmus zu lösen.

## 2.1 Einfache Beschreibung

Um im Vorhinein die Wirkungsweise zu verdeutlichen ohne die dahinter liegende Mathematik anzutasten, beschreiben wir die Methode mithilfe einer Metapher.

- Alice und Bob wollen Nachrichten austauschen. Zu erst möchte Alice an Bob eine Nachricht schicken und legt diese Nachricht in eine Kiste, verschließt sie mit ihrem Schloss und sendet die Kiste an Bob.
- Bob bekommt die Kiste, kann das Schloss von Alice nicht entfernen, da er den Schlüssel hierzu nicht hat. Nun hängt Bob auch sein Schloss an die Truhe und schickt die Truhe wieder zurück an Alice.
- Alice entfernt ihr eigenes Schloss, da sie ursprünglich den Schlüssel hierzu besitzt. Die Kiste ist immer noch verschlossen, da Bobs Schloss noch an der Kiste hängt. Alice schickt nun die Kiste ein letztes mal an Bob.
- Bob bekommt die Kiste mit nur seinem Schloss das er durch seinen eigenen Schlüssel auch wieder entfernen kann. Die Kiste hat nun kein Schloss mehr und Bob kann die Nachricht von Alice lesen die sich in der Kiste befindet.

## 2.2 Voraussetzung

Die Voraussetzung an das Verfahren ist, dass beide über eine gemeinsame und genügend große Primzahl p bescheid wissen. Diese Primzahl kann öffentlich bekannt sein.

Alice und Bob wählen nun jeweils ein Paar von Exponenten $d$ und $e$ mit

$$ed \equiv 1 \ (\mathrm{mod} p-1)$$

, das bedeutet also dass

$$a^{de} \equiv a \ (\mathrm{mod} p-1)$$

für alle ganzen Zahlen $a \in Z$ gilt. Diese Behauptung wurde durch den kleinen Satz von Fermet bereits bestätigt. Diese Voraussetzung ist dann später hilfreich um mit dem Schlüsselpaar die eigene Verschlüsselung wieder aufzuheben. Für die Berechnung von $e$ muss zuerst ein $e$ mit

$$e < p-1 \text{ und } ggT(e, p-1) = 1$$

gefunden werden. $d$ ist dann das multiplikativ Inverse von $e \bmod p-1$.
Beide Parteien Alice und Bob halten ihre Zahlen geheim.

## 2.3 Ablauf

Alice berechnet aus ihrer ursprünglichen Nachricht $m < p$ die Nachricht $x_A$ die sie an Bob sendet

$$x_{A1} = m^{e_A} \bmod p$$

Bob berechnet aus dieser Nachricht seine eigene Rückantwort

$$\begin{aligned} x_B &= (x_A)^{e_B} \bmod p \\ \Leftrightarrow x_B &= (m^{e_A})^{e_B} \bmod p. \end{aligned}$$

Alice erzeugt ausgehend der Rückantwort von Bob und mithilfe ihres Schlüssels nun die nur von Bob verschlüsselte Nachricht.

$$\begin{aligned} x_{A2} &= {x_B}^{d_A} \bmod p \\ \Leftrightarrow x_{A2} &= ((m^{e_A})^{e_B})^{d_A} \bmod p \\ \Leftrightarrow x_{A2} &= ((m^{e_A})^{d_A})^{e_B} \bmod p \\ \Leftrightarrow x_{A2} &= m^{e_B} \bmod p \end{aligned}$$

In diesem Schritt hat Alice also all ihre Verschlüsselung der Ursprungsnachricht wieder aufgehoben und die Nachricht ist nur noch durch die Verschlüsselung von Bob verschlüsselt.
Bob entschlüsselt die Nachricht nun mit

$$x_A 2^{d_B} = (m^{e_B})^{d_B} = m \bmod p.$$

Bob hat damit nun die Ursprungsnachricht von Alice bekommen. Außer Alice und Bob konnte ohne das Wissen der Schlüssel niemand auf die Ursprungsnachricht m während der Datenübertragung zugreifen.

## 2.4 Sicherheitsaspekte

Das Massey-Omura-Schema ist gegen einen Lauschangriff sicher, solange das Problem des Diskreten Logarithmus nicht effizient lösbar ist. Wenn dieser Fall eintreten sollte, ist dieses Verfahren nicht mehr sicher, da aus den einzelnen übertragenen Nachrichten Rückschlüsse auf die Schlüssel gezogen werden könnten.
Einen Wirkungsvollen Schutz gegen einen Man-in-the-Middle Angriff bietet dieses Schema jedoch nicht, da dieses Verfahren nur gegen Lauschangriffe ausgelegt ist.

# 3 Index-Calculus-Methode

Idee:

Wir berechnen den 'großen' Logarithmus, indem wir Logarithmen für 'kleine' Elemente aus $Z_p$ berechnen und damit Rückschlüsse auf den 'großen' Logarithmus ziehen.

## 3.1 Vorberechnung

1. Man bestimmt eine Zahl $B < n$ und die Menge $F(B) = \{p_1, p_2, \cdots, p_B\}$ von Primzahlen, die sogenannte Faktorbasis.

2. Man wähle ein C größer als B, z.B. $B + 10$.

3. Man erhält nun C Kongruenzen der Form $\alpha^{x_j} \equiv {p_1}^{a_{1j}} {p_2}^{a_{2j}} \cdots {p_B}^{a_{Bj}} \mod p$ für $1 \leq j \leq C$.

4. Dies kann man in ein lineares Gleichungssystem mit C Gleichungen umwandeln mit Gleichungen der Form
$x_j \equiv a_{1j} \log_\alpha p_1 + \cdots + a_{Bj} \log_\alpha p_B \mod p - 1$ für $1 \leq j \leq C$.

5. Man bestimmt x-Werte, so dass $\alpha^x$ nur Primfaktoren in $F(B)$ hat, und berechnet die Exponenten der Primfaktoren durch Division.

6. Anschließend löst man das lineare Gleichungssystem mit dem Gauss'schen Algorithmus.

- Die Vorberechnung terminiert in $O(e^{(1+o(1))\sqrt{\ln(p)\ln(\ln(p))}})$.

## 3.2 Bestimmen des diskreten Logarithmus

1. Man wählt zufällig ein $s$ im Bereich $1 \leq s \leq p - 2$ und berechnet $y = \beta\alpha^s \mod p$.

2. Wenn $s$ nur Primfaktoren in $F(B)$ hat, so erhält man
$\beta\alpha^s \equiv p^{c_1 1} p^{c_2 2} \cdots p^{c_B B} \mod p$
ansonsten muss man ein neues $s$ wählen.

3. Jetzt kann man umformen nach
$\log_\alpha \beta + s \equiv c_1 \log_\alpha p_1 \; c_2 \log_\alpha p_2 \; \cdots \; c_B log_\alpha p_B \mod p - 1$.

- Dieser Algorithmus terminiert in $O(e^{(1/2+o(1))\sqrt{\ln(p)\ln(\ln(p))}})$.

### 3.3 Beispiel

Wir nehmen an, dass $p = 83$ ist. Wir bestimmen nun $B = 7$ und $F(B) = p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7$ als die Faktorbasis aus Primzahlen. Wir wählen außerdem ein $C = 17$.

In folgender Tabelle betrachten wir alles mit mod 82.

$$\begin{aligned}
2^1 &\equiv 2 \\
2^7 &\equiv 45 = 3^2 \cdot 5 \\
2^8 &\equiv 7 \\
2^9 &\equiv 14 = 2 \cdot 7 \\
2^{10} &\equiv 28 = 2^2 \cdot 7 \\
2^{11} &\equiv 56 = 2^3 \cdot 7 \\
2^{12} &\equiv 29 \\
2^{13} &\equiv 58 = 2 \cdot 29 \\
2^{14} &\equiv 33 = 3 \cdot 11 \\
2^{15} &\equiv 66 = 3 \cdot 2 \cdot 11 \\
2^{16} &\equiv 49 = 7^2 \\
2^{17} &\equiv 15 = 3 \cdot 5
\end{aligned}$$

Aus diesem System nehmen wir nur diejenigen Gleichungen heraus, die wir durch unsere vorher bestimmte Faktorbasis darstellen können. In unserem Fall $2^1, 2^7, 2^8, 2^9$ *und* $2^{17}$. Wir hätten auch mehr zur Verfügung, aber diese Anzahl reicht bereits.

Die Gleichungen geben uns ein lineares Gleichungssystem das wir lösen. Damit sind unsere Vorbereitungen abgeschlossen und wir bekommen:

$$\log_2(2) = 1,\ \log_2(3) = 72,\ \log_2(5) = 27,\ \log_2(7) = 8$$

| 2 | 3 | 5 | 7 | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |
| 0 | 2 | 1 | 0 | 7 |
| 0 | 0 | 0 | 1 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 0 | 1 | 1 | 0 | 17 |
| 0 | 1 | 0 | 0 | - 10 = 72 |
| 0 | 0 | 1 | 0 | 34 - 7 = 27 |

Wenn wir nun $\log_2(31)$ wissen möchten gehen wir wie folgt vor.

$$\begin{aligned} 31^2 &\equiv 48 = 2^4 \cdot 3 \quad |log \\ 2 \cdot \log_2(31) &\equiv 4 \cdot \log_2 2 + \log_2 3 \\ \Longleftrightarrow 2 \cdot \log_2(31) &\equiv 1 + 1 + 1 + 1 + 72 = 76 \\ daraus folgt: & \\ \log_2(31) &\equiv 38 \; oder \\ \log_2(31) &\equiv 38 + 41 = 79 \end{aligned}$$

und wie wir aus unserer Annahme leicht nachprüfen können ist $2^{38} \bmod 83 = 31$. Der wert 38 + 41 ist durch das Modulorechnen und die Teilung durch 2 zu überprüfen und mit einzuplanen.

### 3.4 Bemerkung

Von einem Laufzeitstandpunkt aus gesehen ist bei großen Primzahlen das Index-Calculus-Verfahren schneller als z.b. Shanks Baby-Step-Giant-Step Algorithmus. Jedoch kann man nicht davon sprechen dass der diskrete Logarithmus hiermit schnell im Vergleich zur diskreten Exponentiation berechnet werden kann.